2023

Stocker (Stock Market Prediction)

NES RATNAM COLLEGE OF ARTS, SCIENCE AND COMMERCE

Created By -Aniket Vijaybahadur Vishwakarma
TYBsc Computer Science
Roll No: 02

'A' GRADE
NAAC Re-accredited

NATIONAL EDUCATION SOCIETY'S

Ratnam College of Arts, Science & Commerce
Bhandup, Mumbai – 400 078.

Department of Computer Science

CERTIFICATE

Class: ***Roll / Seat No.:*** ***Year:***

This is to certify that Mr./Ms.________________________has satisfactorily completed the project work___________________for partial fulfillment of the 3 years Degree Course Bachelor in Computer Science for the University of Mumbai for the year ________to________.

Place:________________

Date:________________

____
Project Guide

____
In-charge,
Dept. of Computer Science

____
Signature of Examiner with Date

ACKNOWLEDGEMENT

Apart from the efforts of team, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project. The completion of any inter-disciplinary project depends upon cooperation, co-ordination and combined efforts of several sources of knowledge. We are eternally grateful to our teacher Arti Debnath for her even willingness to give us valuable advice and direction under which we executed this project. Her constant guidance and willingness to share her vast knowledge made us understand this project and its manifestations in great depths and helped us to complete the assigned task

Table of content

1. Introduction

What is Stock Market?

Stock market prediction is the act of trying to determine the future value of a company stock or other financial instrument traded on an exchange. The successful prediction of a stock's future price could yield significant profit.

The efficient-market hypothesis suggests that stock prices reflect all currently available information and any price changes that are not based on newly revealed information thus are inherently unpredictable. Others disagree and those with this viewpoint possess myriad methods and technologies which purportedly allow them to gain future price information

Importance of Stock Market

- Stock markets help companies to raise capital.
- It helps generate personal wealth.
- Stock markets serve as an indicator of the state of the economy.
- It is a widely used source for people to invest money in companies with high growth potential.

Stock Price Prediction

Stock Price Prediction using machine learning helps you discover the future value of company stock and other financial assets traded on an exchange. The entire idea of predicting stock prices is to gain significant profits. Predicting how the stock market will perform is a hard task to do. There are other factors involved in the prediction, such as physical and psychological factors, rational and irrational behaviour, and so on. All these factors combine to make share prices dynamic and volatile. This makes it very difficult to predict stock prices with high accuracy.

Why we need Stock Market Prediction Software?

The majority of people turn to the performance of a country's stock market as the best indicator of how well that economy is doing. Stock markets cover all industries across all sectors of the economy. This means they serve as a barometer of what cycle the economy is in and the hopes and fears of the population who generate growth and wealth.

Stock markets have existed for centuries and will no doubt go on being the main public, regulated marketplaces where people can buy and sell shares of different companies.

Of course, today's markets are very different from share trading in the Dutch East India Company back in 1602, but stocks still remain the most popular investment choice thanks to their potential for returns and their opportunity to invest directly in individual companies.

Stock market prediction and analysis are some of the most difficult jobs to complete. There are numerous causes for this, including market volatility and a variety of other dependent and independent variables that influence the value of a certain stock in the market. These variables make it extremely difficult for any stock market expert to anticipate the rise and fall of the market with great precision.

Problem Statement for Stock Market Prediction

Let us see the data on which we will be working before we begin implementing the software to anticipate stock market values. In this section, we will examine the stock price of Microsoft Corporation (MSFT) as

reported by the National Association of Securities Dealers Automated Quotations (NASDAQ). The stock price data will be supplied as a Comma Separated File (.csv) that may be opened and analyzed in Excel or a Spreadsheet.
MSFT's stocks are listed on NASDAQ, and their value is updated every working day of the stock market. It should be noted that the market does not allow trading on Saturdays and Sundays. Therefore, there is a gap between the two dates. The Opening Value of the stock, the Highest and Lowest values of that stock on the same day, as well as the Closing Value at the end of the day are all indicated for each date.

The Adjusted Close Value reflects the stock's value after dividends have been declared (too technical!). Furthermore, the total volume of the stocks in the market is provided. With this information, it is up to the job of a Machine Learning/Data Scientist to look at the data and develop different algorithms that may extract patterns from the historical data of the Microsoft Corporation stock.

2. Software Requirement Specification

Stocker (Stock market prediction) is a software which helps users to know about which company stock will be increasing or decreasing. It uses 10 years of previous data to get accurate result. It uses yahoo financial to get the data. Stocker is easy to access and free to use to predict the stock market, the website is so simple to use for everyone with simple user-friendly user interface.
We will use the Long Short-Term Memory (LSTM) method to create a Machine Learning model to forecast Microsoft Corporation stock values. They are used to make minor changes to the information by multiplying and adding. Long-term memory (LSTM) is a deep learning artificial recurrent neural network (RNN) architecture.
Stock market prediction is the act of trying to determine the future value of a company stock or other financial instrument traded on an exchange. The successful prediction of a stock's future price could yield significant profit.

Functional requirements:
The Stock Market Prediction delivers many features like:

- Ability to preprocess the data to remove irrelevant content and noise.
- Implementation of machine learning models to know the stock market price
- User interface to display results to the user.

Non-Functional requirements:
The Software delivers some non-functional features like:

- It is highly scalable and is able to handle increased traffic and data volumes in the future.
- It is highly available and reliable with minimal downtime and errors.
- It is easy to maintain and errors and bugs are fixed easily.

- It handles a large volume of data and requests without any significant delays.

User Interface:
User interface design, layout, navigation and interactions:

- A very clean and intuitive interface that allows user to easily view news articles and detect fake news and reals news.
- A very user-friendly website to easily interact with the user and the users can easily understand the website navigation.

System Interface:

- It uses streamlit library for GUI in python to run in localhost and TensorFlow to train model, yfinance to load the data from yahoo finance website

Hardware :
The hardware requirements for a fake news detection website using Python will depend on the scale of the website and the complexity of the machine learning algorithms used for fake news detection. Here are some general recommendations for a small to medium-scale website:

- Processor: A multicore processor with a clock speed of at least 2 GHz is recommended. A higher clock speed and more cores will result in faster processing of large datasets and more complex machine learning algorithms.
- RAM: A minimum of 8 GB of RAM is recommended for running machine learning algorithms and web server applications. However, for larger datasets and more complex algorithms, 16 GB or 32 GB of RAM may be required.
- Storage: A solid-state drive (SSD) is recommended for faster access to data and faster processing of machine learning algorithms. A minimum of 256 GB of storage is recommended.
- Graphics processing unit (GPU): A GPU can accelerate machine learning algorithms, especially those that involve deep learning. A mid-range GPU with at least 4 GB of VRAM is recommended.

3. Software

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0.[36] Python 2.0 was released in 2000. Python 3.0, released in 2008, was a major revision not completely backward-compatible with earlier versions. Python 2.7.18, released in 2020, was the last release of Python 2.[37]

Python consistently ranks as one of the most popular programming languages

Library Used:

1) TensorFlow

TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications.

2) Scikit-learn

scikit-learn is a free software machine learning library for the Python programming language
It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. Scikit-learn is a NumFOCUS fiscally sponsored project.

3) NumPy

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

4) Keras

Keras is an open-source high-level Neural Network library, which is written in Python is capable enough to run on Theano, TensorFlow, or CNTK. It was developed by one of the Google engineers, Francois Chollet. It is made user-friendly, extensible, and modular for facilitating faster experimentation with deep neural networks. It not only supports Convolutional Networks and Recurrent Networks individually but also their combination.

5) Pandas

Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas is fast and it has high performance & productivity for users.

6) Matplotlib

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

7) Streamlit

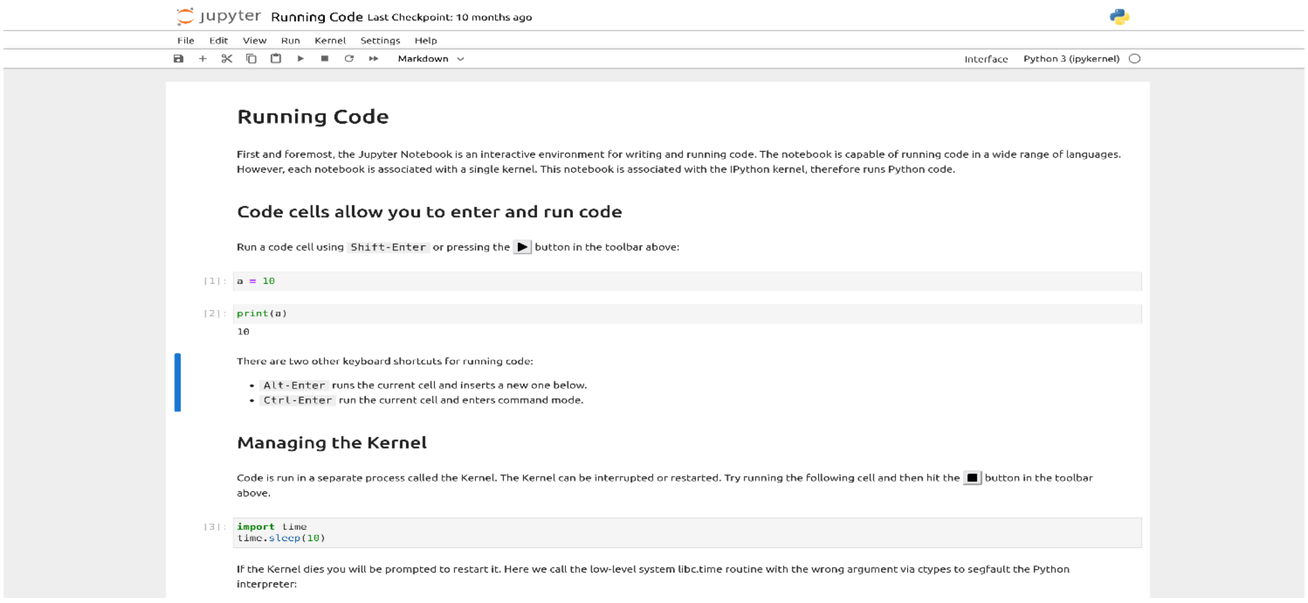
Streamlit is a free and open-source framework to rapidly build and share beautiful machine learning and data science web apps. It is a Python-based library specifically designed for machine learning engineers.

8) Yfinance

yfinance is one of the famous modules in Python, which is used to collect online data, and with it, we can collect the financial data of Yahoo. With the help of the yfinance module, we retrieve and collect the company's financial information (such as financial ratios, etc.) as well as the histories of marketing data by using its functions.

Tools:

Jupyter


jupyter Running Code Last Checkpoint: 10 months ago
File Edit View Run Kernel Settings Help
Markdown
Interface Python 3 (ipykernel)

Running Code

First and foremost, the Jupyter Notebook is an interactive environment for writing and running code. The notebook is capable of running code in a wide range of languages. However, each notebook is associated with a single kernel. This notebook is associated with the IPython kernel, therefore runs Python code.

Code cells allow you to enter and run code

Run a code cell using `Shift-Enter` or pressing the ▶ button in the toolbar above:

```
[1]: a = 10
```

```
[2]: print(a)
10
```

There are two other keyboard shortcuts for running code:

- `Alt-Enter` runs the current cell and inserts a new one below.
- `Ctrl-Enter` run the current cell and enters command mode.

Managing the Kernel

Code is run in a separate process called the Kernel. The Kernel can be interrupted or restarted. Try running the following cell and then hit the ■ button in the toolbar above.

```
[3]: import time
     time.sleep(10)
```

If the Kernel dies you will be prompted to restart it. Here we call the low-level system libc.time routine with the wrong argument via ctypes to segfault the Python interpreter:

Jupyter notebook is a language-agnostic HTML notebook application for Project Jupyter. In 2015, Jupyter notebook was released as a part of The Big Split of the IPython codebase. IPython 3 was the last major monolithic release containing both language-agnostic code, such as the *IPython notebook*, and language specific code, such as the *IPython kernel for Python*. As computing spans across many languages, Project Jupyter will continue to develop the language-agnostic Jupyter notebook in this repo and with the help of the community develop language specific kernels which are found in their own discrete repos.

Installation

You can find the installation documentation for the Jupyter platform, on ReadTheDocs. The documentation for advanced usage of Jupyter notebook can be found here.

For a local installation, make sure you have pip installed and run:

```
pip install notebook
```

Visual Studio Code

Visual Studio Code is a source-code editor that can be used with a variety of programming languages,
including C, C#, C++, Fortran, Go, Java, JavaScript, Node.js, Python, Rust. It is based on the Electron framework, which is used to develop Node.js web applications that run on the Blink layout engine. Visual Studio Code employs the same editor component (codenamed "Monaco") used in Azure DevOps
Visual Studio Code, also commonly referred to as VS Code,[9] is a source-code editor made by Microsoft with the Electron Framework , for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add functionality.

Installation:

https://code.visualstudio.com/download

4. System Design Details

Methodology: The methodology for building a fake news detection website using Python typically involves the following steps:

1) Collecting Data:

As you know, machines initially learn from the data that you give them. It is of the utmost importance to collect reliable data so that your machine learning model can find the correct patterns. The quality of the data that you feed to the machine will determine how accurate your model is. If you have incorrect or outdated data, you will have wrong outcomes or predictions which are not relevant

Make sure you use data from a reliable source, as it will directly affect the outcome of your model. Good data is relevant, contains very few missing and repeated values, and has a good representation of the various subcategories/classes present.

2) Preparing the Data:

After you have your data, you have to prepare it. You can do this by :

- Putting together all the data you have and randomizing it. This helps make sure that data is evenly distributed, and the ordering does not affect the learning process.
- Cleaning the data to remove unwanted data, missing values, rows, and columns, duplicate values, data type conversion, etc. You might even have to restructure the dataset and change the rows and columns or index of rows and columns.
- Visualize the data to understand how it is structured and understand the relationship between various variables and classes present.

- Splitting the cleaned data into two sets - a training set and a testing set. The training set is the set your model learns from. A testing set is used to check the accuracy of your model after training.

3) Choosing a Model:

A machine learning model determines the output you get after running a machine learning algorithm on the collected data. It is important to choose a model which is relevant to the task at hand. Over the years, scientists and engineers developed various models suited for different tasks like speech recognition, image recognition, prediction, etc. Apart from this, you also have to see if your model is suited for numerical or categorical data and choose accordingly.

4) Training the Model:

Training is the most important step in machine learning. In training, you pass the prepared data to your machine learning model to find patterns and make predictions. It results in the model learning from the data so that it can accomplish the task set. Over time, with training, the model gets better at predicting.

5) Evaluating the Model:

After training your model, you have to check to see how it's performing. This is done by testing the performance of the model on previously unseen data. The unseen data used is the testing set that you split our data into earlier. If testing was done on the same data which is used for training, you will not get an accurate measure, as the model is already used to the data, and finds the same patterns in it, as it previously did. This will give you disproportionately high accuracy.

When used on testing data, you get an accurate measure of how your model will perform and its speed.

6) Parameter Tuning:

Once you have created and evaluated your model, see if its accuracy can be improved in any way. This is done by tuning the parameters present in your model. Parameters are the variables in the model that the programmer generally decides. At a particular value of your parameter, the accuracy will be the maximum. Parameter tuning refers to finding these values.

7) Making Predictions

In the end, you can use your model on unseen data to make predictions accurately.

What is LSTM(Long Short Term Memory) Model?

Long Short-Term Memory Networks is a deep learning, sequential neural network that allows information to persist. It is a special type of Recurrent Neural Network which is capable of handling the vanishing gradient problem faced by RNN. LSTM was designed by Hochreiter and Schmidhuber that resolves the problem caused by traditional rnns and machine learning algorithms. LSTM can be implemented in Python using the Keras library.

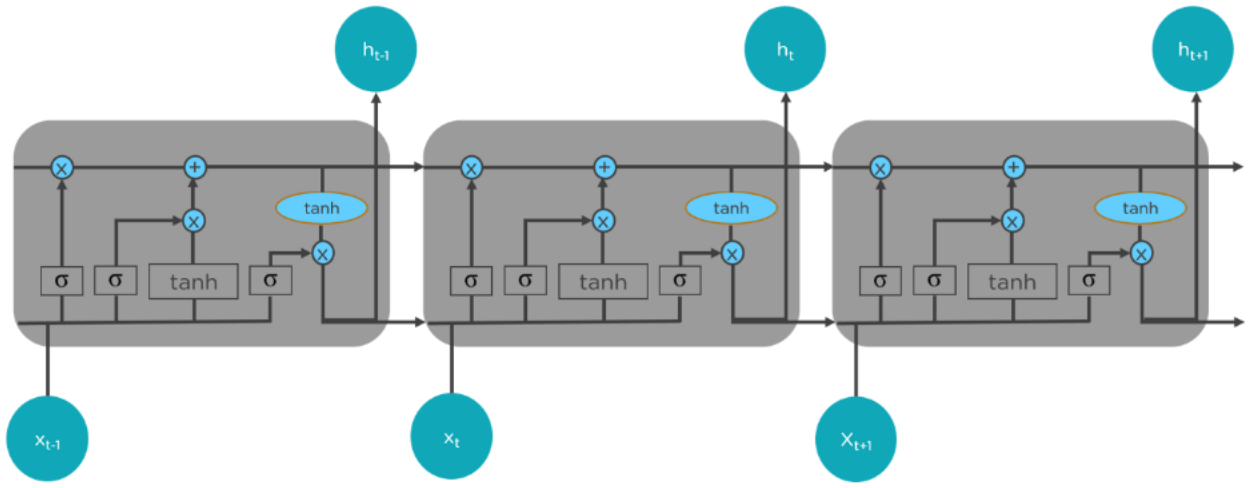
Let’s say while watching a video, you remember the previous scene, or while reading a book, you know what happened in the earlier chapter. RNNs work similarly; they remember the previous information and use it for processing the current input. The shortcoming of RNN is they cannot remember long-term dependencies due to vanishing gradient. LSTMs are explicitly designed to avoid long-term dependency problems.

This article will cover all the basics about LSTM, including its meaning, architecture, applications, and gates.

Understanding Long Short Term Memory Network

Here, you will use a Long Short Term Memory Network (LSTM) for building your model to predict the stock prices of Google.

LTSMs are a type of Recurrent Neural Network for learning long-term dependencies. It is commonly used for processing and predicting time-series data.



From the image on the top, you can see LSTMs have a chain-like structure. General RNNs have a single neural network layer. LSTMs, on the other hand, have four interacting layers communicating extraordinarily.

LSTMs work in a three-step process.

- The first step in LSTM is to decide which information to be omitted from the cell in that particular time step. It is decided with the help of a sigmoid function. It looks at the previous state (ht-1) and the current input xt and computes the function.
- There are two functions in the second layer. The first is the sigmoid function, and the second is the tanh function. The sigmoid function decides which values to let through (0 or 1). The tanh function gives the weightage to the values passed, deciding their level of importance from -1 to 1.

- The third step is to decide what will be the final output. First, you need to run a sigmoid layer which determines what parts of the cell state make it to the output. Then, you must put the cell state through the tanh function to push the values between -1 and 1 and multiply it by the output of the sigmoid gate.

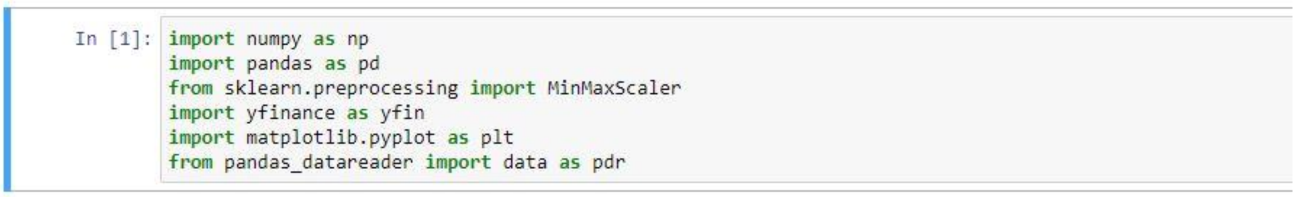
With this basic understanding of LSTM, you can dive into the hands-on demonstration part of this tutorial regarding stock price prediction using machine learning.

5. System Implementation

Yahoo Finance Stock Price Prediction Using LSTM:

1. Import the Libraries.

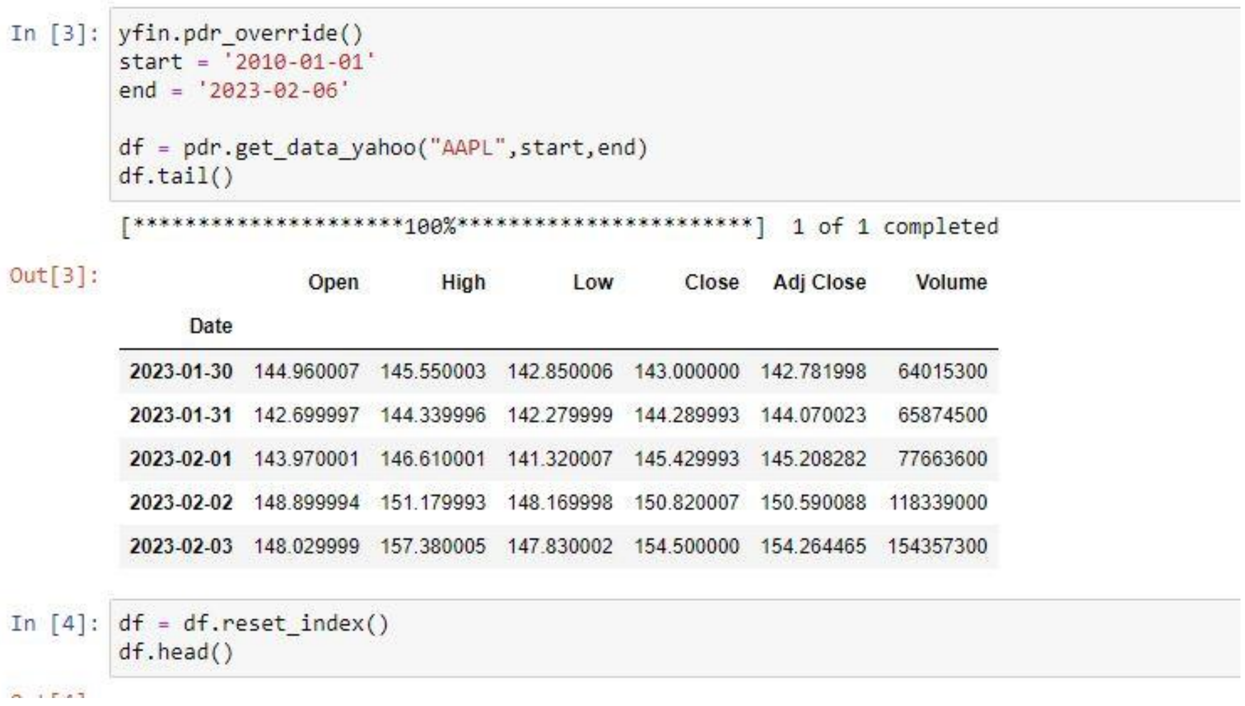
To begin, we include all of the libraries used for this project. I used the yfinance API to gather all of the historical stock market data. It's taken directly from the yahoo finance website, so it's very reliable data.



```
In [1]: import numpy as np
        import pandas as pd
        from sklearn.preprocessing import MinMaxScaler
        import yfinance as yfin
        import matplotlib.pyplot as plt
        from pandas_datareader import data as pdr
```

2. Load the Training Dataset.

The Google training data has information from 3 Jan 2012 to 30 Dec 2016. There are five columns. The Open column tells the price at which a stock



```
In [3]: yfin.pdr_override()
        start = '2010-01-01'
        end = '2023-02-06'

        df = pdr.get_data_yahoo("AAPL",start,end)
        df.tail()

        [*********************100%***********************]  1 of 1 completed
Out[3]:
```

Date	Open	High	Low	Close	Adj Close	Volume
2023-01-30	144.960007	145.550003	142.850006	143.000000	142.781998	64015300
2023-01-31	142.699997	144.339996	142.279999	144.289993	144.070023	65874500
2023-02-01	143.970001	146.610001	141.320007	145.429993	145.208282	77663600
2023-02-02	148.899994	151.179993	148.169998	150.820007	150.590088	118339000
2023-02-03	148.029999	157.380005	147.830002	154.500000	154.264465	154357300

```
In [4]: df = df.reset_index()
        df.head()
```

started trading when the market opened on a particular day.

The Close column refers to the price of an individual stock when the stock exchange closed the market for the day. The High column depicts the highest price at which a stock traded during a period. The Low column tells the lowest price of the period. Volume is the total amount of trading activity during a period of time.

3. Data Processing & Feature Engineering

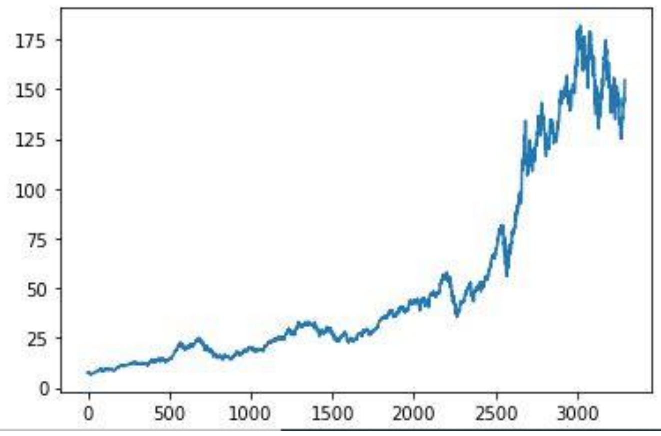
We see that our data above is rough and contains lots of spikes for a time series. It isn’t very smooth and can be difficult for the model to extract trends from. To reduce the appearance of this we want to exponentially smooth our data before we compute the technical indicators.

```
In [5]: df= df.drop(['Date', 'Adj Close'], axis=1)
        df.head()
```

Out[5]:

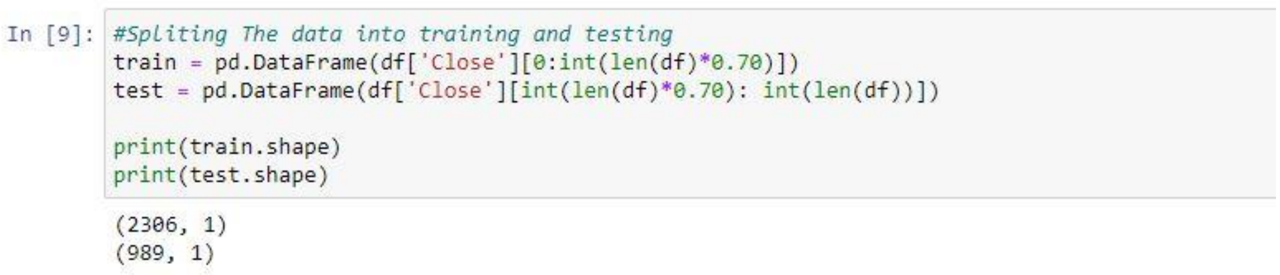
	Open	High	Low	Close	Volume
0	7.622500	7.660714	7.585000	7.643214	493729600
1	7.664286	7.699643	7.616071	7.656429	601904800
2	7.656429	7.686786	7.526786	7.534643	552160000
3	7.562500	7.571429	7.466071	7.520714	477131200
4	7.510714	7.571429	7.466429	7.570714	447610800

```
In [6]: plt.plot(df.Close)
Out[6]: [<matplotlib.lines.Line2D at 0x1ebac27f790>]
```



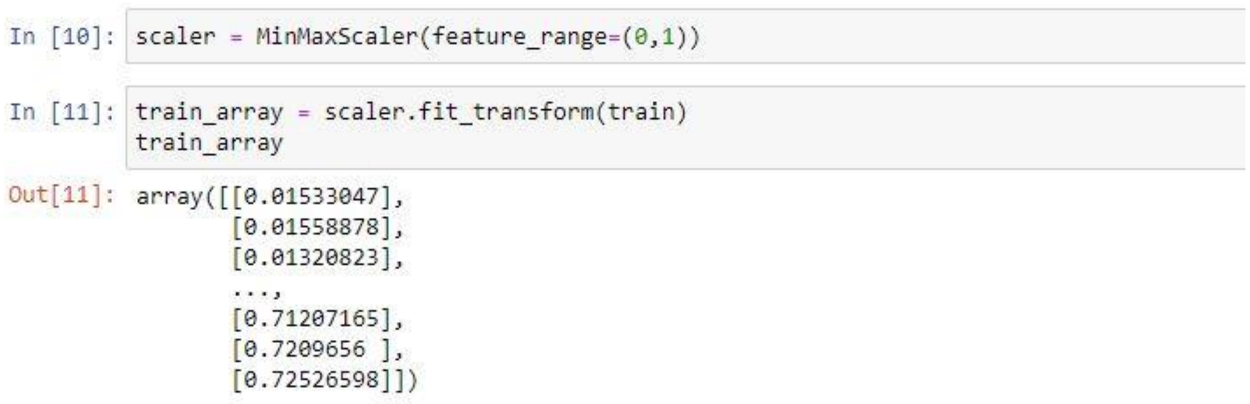
I remove the columns like ‘Date’ and ‘Adj Close’ because we can get a good enough approximation with our ema’s in addition to the indicators. Volume has been proven to have a correlation with price fluctuations, which is why I normalized it.

4. Use the Open Stock Price Column to Train Your Model.



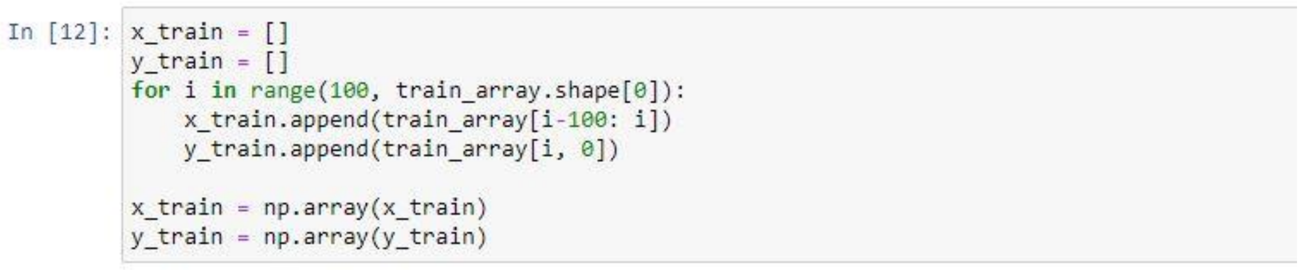
```
In [9]: #Spliting The data into training and testing
        train = pd.DataFrame(df['Close'][0:int(len(df)*0.70)])
        test = pd.DataFrame(df['Close'][int(len(df)*0.70): int(len(df))])

        print(train.shape)
        print(test.shape)

        (2306, 1)
        (989, 1)
```

5. Normalizing the Dataset.



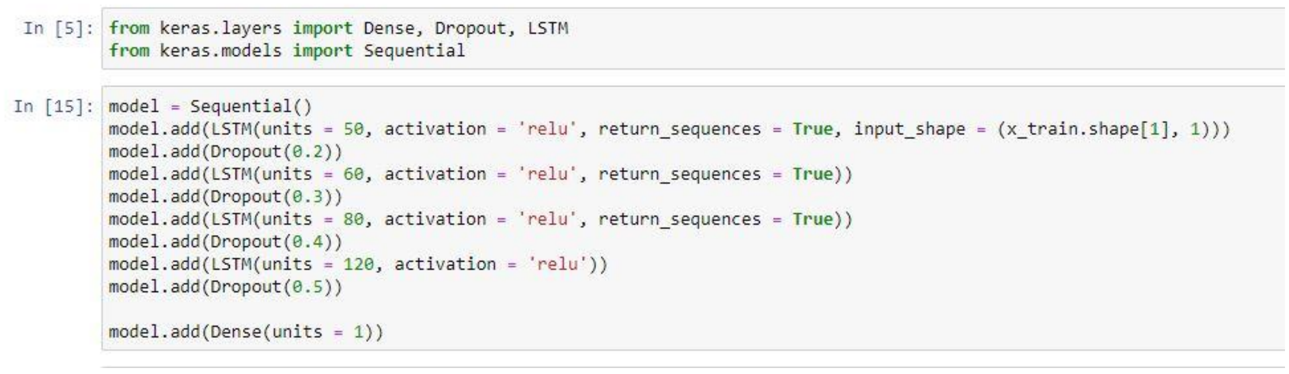
```
In [10]: scaler = MinMaxScaler(feature_range=(0,1))

In [11]: train_array = scaler.fit_transform(train)
         train_array

Out[11]: array([[0.01533047],
                [0.01558878],
                [0.01320823],
                ...,
                [0.71207165],
                [0.7209656 ],
                [0.72526598]])
```

6. Creating X_train and y_train Data Structures.



```
In [12]: x_train = []
         y_train = []
         for i in range(100, train_array.shape[0]):
             x_train.append(train_array[i-100: i])
             y_train.append(train_array[i, 0])

         x_train = np.array(x_train)
         y_train = np.array(y_train)
```

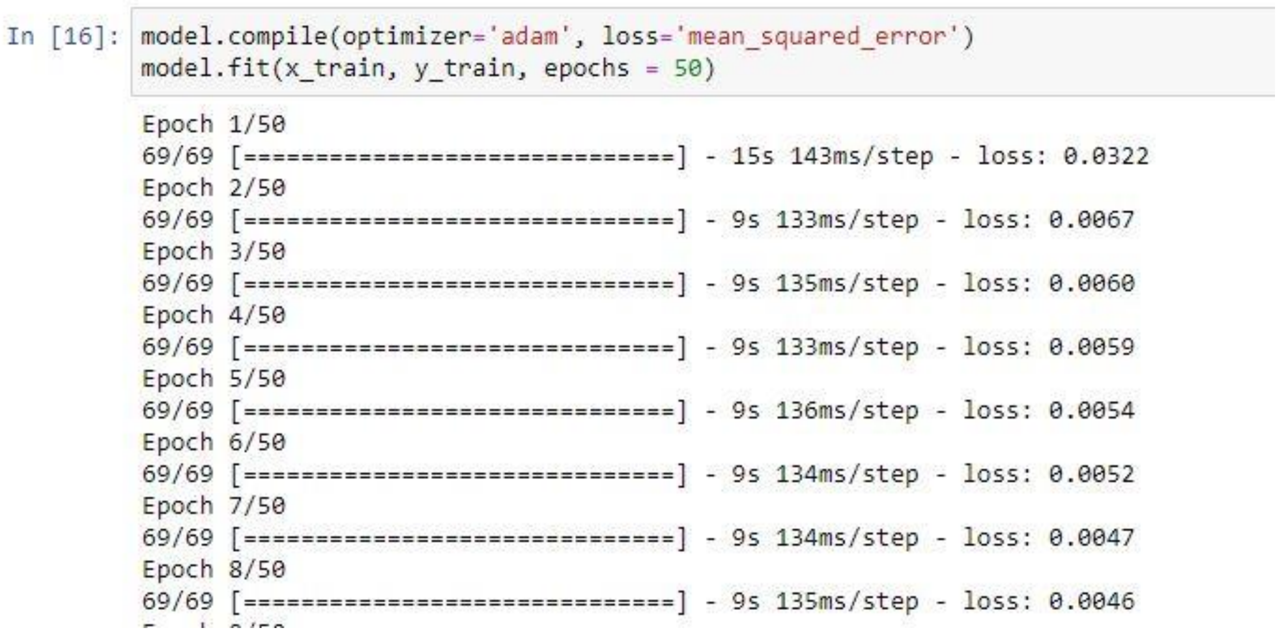
7. Building the Model by Importing the Crucial Libraries and Adding Different Layers to LSTM.



```
In [5]: from keras.layers import Dense, Dropout, LSTM
        from keras.models import Sequential
```

```
In [15]: model = Sequential()
         model.add(LSTM(units = 50, activation = 'relu', return_sequences = True, input_shape = (x_train.shape[1], 1)))
         model.add(Dropout(0.2))
         model.add(LSTM(units = 60, activation = 'relu', return_sequences = True))
         model.add(Dropout(0.3))
         model.add(LSTM(units = 80, activation = 'relu', return_sequences = True))
         model.add(Dropout(0.4))
         model.add(LSTM(units = 120, activation = 'relu'))
         model.add(Dropout(0.5))

         model.add(Dense(units = 1))
```

8. Fitting the Model.



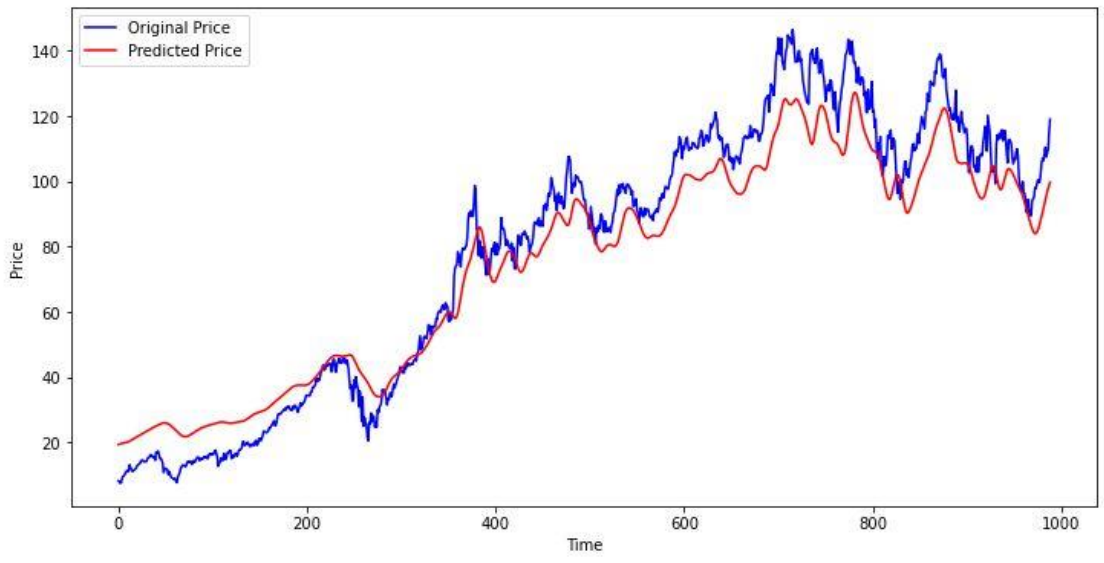
```
In [16]: model.compile(optimizer='adam', loss='mean_squared_error')
         model.fit(x_train, y_train, epochs = 50)

         Epoch 1/50
         69/69 [==============================] - 15s 143ms/step - loss: 0.0322
         Epoch 2/50
         69/69 [==============================] - 9s 133ms/step - loss: 0.0067
         Epoch 3/50
         69/69 [==============================] - 9s 135ms/step - loss: 0.0060
         Epoch 4/50
         69/69 [==============================] - 9s 133ms/step - loss: 0.0059
         Epoch 5/50
         69/69 [==============================] - 9s 136ms/step - loss: 0.0054
         Epoch 6/50
         69/69 [==============================] - 9s 134ms/step - loss: 0.0052
         Epoch 7/50
         69/69 [==============================] - 9s 134ms/step - loss: 0.0047
         Epoch 8/50
         69/69 [==============================] - 9s 135ms/step - loss: 0.0046
```

9. Predicting the Values of testing data

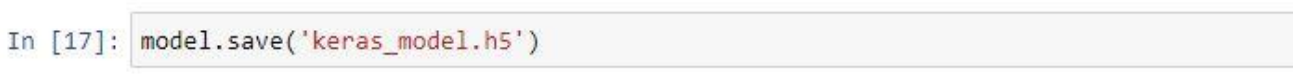
```
In [36]: #Makng Prediction

In [37]: y_pred = model.predict(x_test)

         31/31 [==============================] - 2s 41ms/step

In [38]: y_pred.shape

Out[38]: (989, 1)

In [39]: scaler.scale_

Out[39]: array([0.00682769])

In [40]: scale_factor = 1/0.00682769
         y_pred = y_pred*scale_factor
         y_test = y_test*scale_factor
```

10. Plotting the Actual and Predicted Prices

```
In [41]: plt.figure(figsize=(12,6))
         plt.plot(y_test, 'b', label = 'Original Price')
         plt.plot(y_pred, 'r', label = 'Predicted Price')
         plt.xlabel('Time')
         plt.ylabel('Price')
         plt.legend()
         plt.show()
```



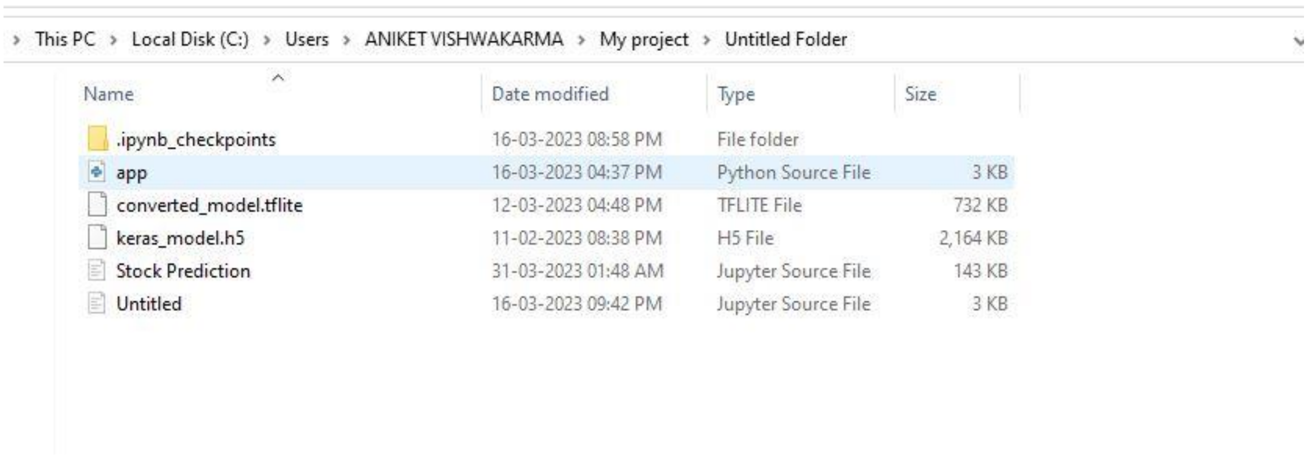
11. Save it as “keras_model.h5”



```
In [17]: model.save('keras_model.h5')
```

How to run app.py?

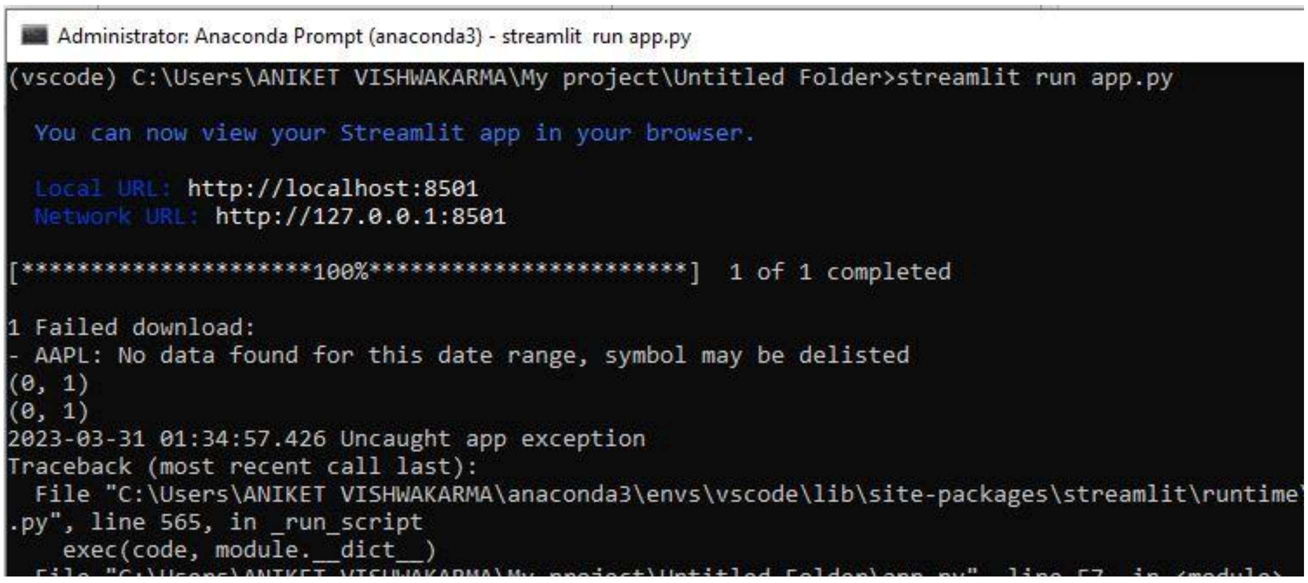
1. First open the folder where project
2. In path box type cmd



› This PC › Local Disk (C:) › Users › ANIKET VISHWAKARMA › My project › Untitled Folder

Name	Date modified	Type	Size
.ipynb_checkpoints	16-03-2023 08:58 PM	File folder	
app	16-03-2023 04:37 PM	Python Source File	3 KB
converted_model.tflite	12-03-2023 04:48 PM	TFLITE File	732 KB
keras_model.h5	11-02-2023 08:38 PM	H5 File	2,164 KB
Stock Prediction	31-03-2023 01:48 AM	Jupyter Source File	143 KB
Untitled	16-03-2023 09:42 PM	Jupyter Source File	3 KB

3. It will open command prompt and in command prompt type “streamlit run app.py”



Administrator: Anaconda Prompt (anaconda3) - streamlit run app.py

```
(vscode) C:\Users\ANIKET VISHWAKARMA\My project\Untitled Folder>streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://127.0.0.1:8501

[*********************100%***********************]  1 of 1 completed

1 Failed download:
- AAPL: No data found for this date range, symbol may be delisted
(0, 1)
(0, 1)
2023-03-31 01:34:57.426 Uncaught app exception
Traceback (most recent call last):
  File "C:\Users\ANIKET VISHWAKARMA\anaconda3\envs\vscode\lib\site-packages\streamlit\runtime
.py", line 565, in _run_script
    exec(code, module.__dict__)
```

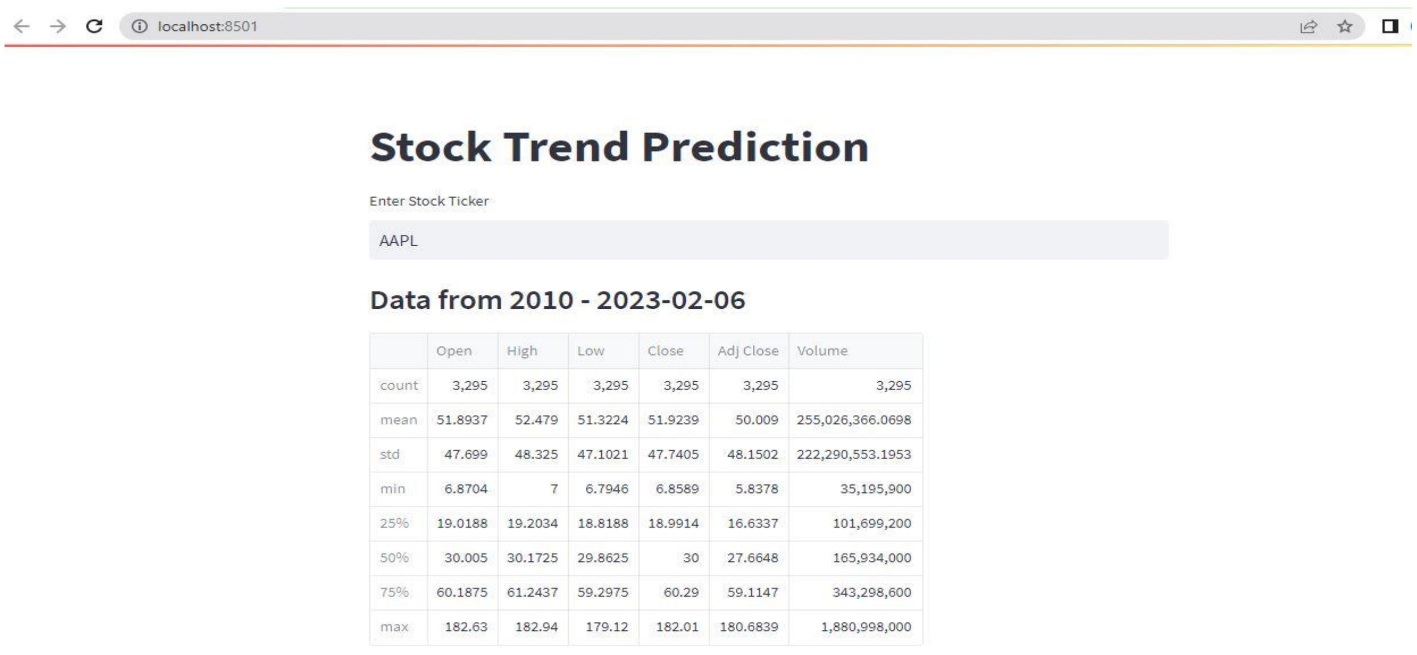
6. CODE

App.py

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import yfinance as yfin
import matplotlib.pyplot as plt
from pandas_datareader import data as pdr
import streamlit as st
from keras.layers import Dense, Dropout, LSTM
from keras.models import Sequential
from keras import models



yfin.pdr_override()
start = '2010-01-01'
end = '2023-02-06'

st.title('Stock Trend Prediction')

input = st.text_input('Enter Stock Ticker', 'AAPL')
df = pdr.get_data_yahoo(input,start,end)
df.tail()

#Describing Data
st.subheader('Data from 2010 - '+end)
st.write(df.describe())

st.subheader('Closing Price vs Time chart')
fig = plt.figure(figsize = (12,6))
plt.plot(df.Close, 'b')
st.pyplot(fig)
st.subheader('Closing Price vs Time chart with 100MA')
MA100 = df.Close.rolling(100).mean()
fig = plt.figure(figsize = (12,6))
plt.plot(MA100 , 'r')
plt.plot(df.Close ,'b')
st.pyplot(fig)

st.subheader('Closing Price vs Time chart with 100MA & 200MA')
MA100 = df.Close.rolling(100).mean()
MA200 = df.Close.rolling(200).mean()
fig = plt.figure(figsize = (12,6))
plt.plot(MA100 ,'r')
plt.plot(MA200, 'g')
plt.plot(df.Close, 'b')
st.pyplot(fig)

#Spliting The data into training and testing
train = pd.DataFrame(df['Close'][0:int(len(df)*0.70)])
test = pd.DataFrame(df['Close'][int(len(df)*0.70): int(len(df))])

print(train.shape)
print(test.shape)

scaler = MinMaxScaler(feature_range=(0,1))
train_array = scaler.fit_transform(train)


model = models.load_model('keras_model.h5')

past_100_days = train.tail(100)
final_df = past_100_days.append(test, ignore_index=True)
input_data = scaler.fit_transform(final_df)

x_test = []
y_test = []
for i in range(100, input_data.shape[0]):
   x_test.append(input_data[i-100: i])
   y_test.append(input_data[i, 0])

x_test, y_test = np.array(x_test), np.array(y_test)
```

```
y_pred = model.predict(x_test)

scaler_val = scaler.scale_
scale_factor = 1/scaler_val[0]
y_pred = y_pred*scale_factor
y_test = y_test*scale_factor

st.subheader('Prediction vs Original')
fig2 = plt.figure(figsize=(12,6))
plt.plot(y_test, 'b', label = 'Original Price')
plt.plot(y_pred, 'r', label = 'Predicted Price')
plt.xlabel('Time')
plt.ylabel('Price')
plt.legend()
st.pyplot(fig2)
```

7. Screenshot



localhost:8501

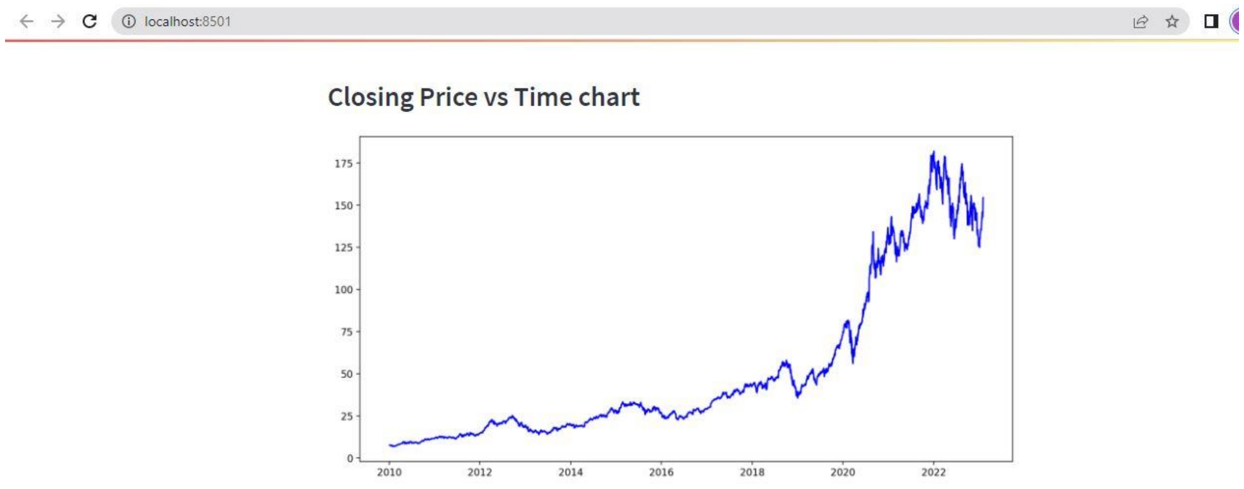
Stock Trend Prediction

Enter Stock Ticker

AAPL

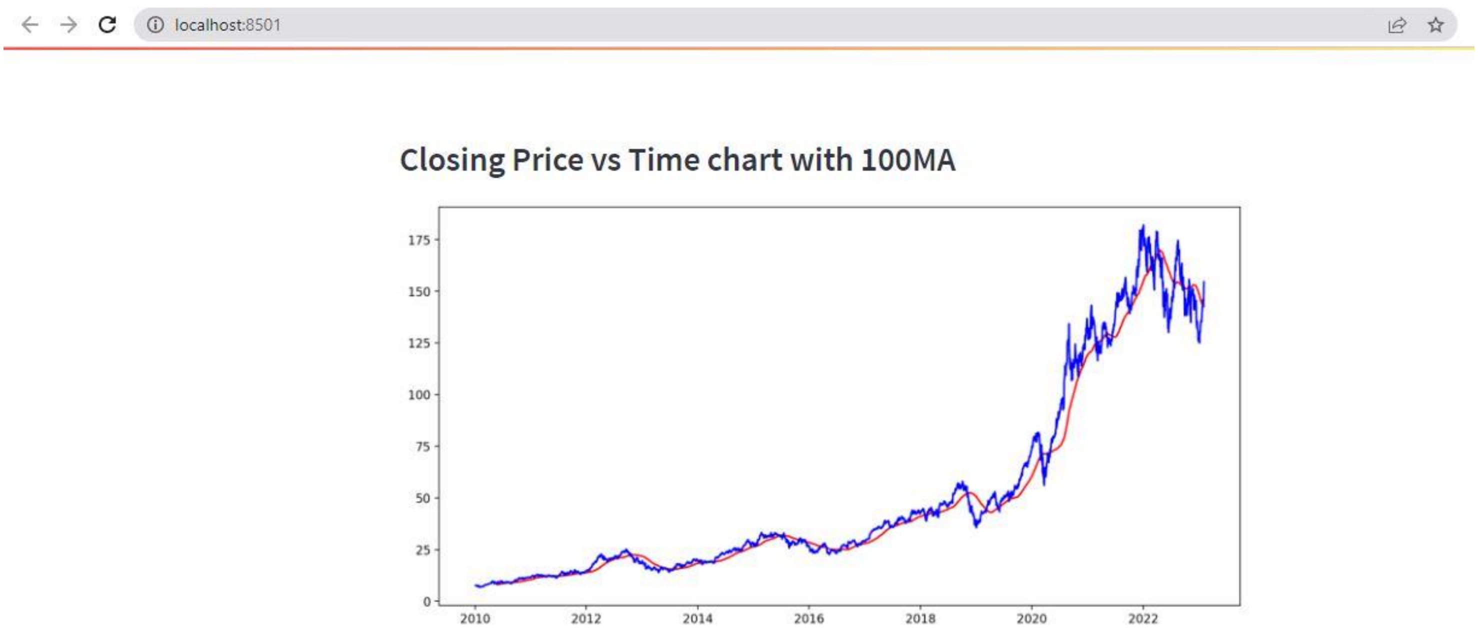
Data from 2010 - 2023-02-06

	Open	High	Low	Close	Adj Close	Volume
count	3,295	3,295	3,295	3,295	3,295	3,295
mean	51.8937	52.479	51.3224	51.9239	50.009	255,026,366.0698
std	47.699	48.325	47.1021	47.7405	48.1502	222,290,553.1953
min	6.8704	7	6.7946	6.8589	5.8378	35,195,900
25%	19.0188	19.2034	18.8188	18.9914	16.6337	101,699,200
50%	30.005	30.1725	29.8625	30	27.6648	165,934,000
75%	60.1875	61.2437	59.2975	60.29	59.1147	343,298,600
max	182.63	182.94	179.12	182.01	180.6839	1,880,998,000

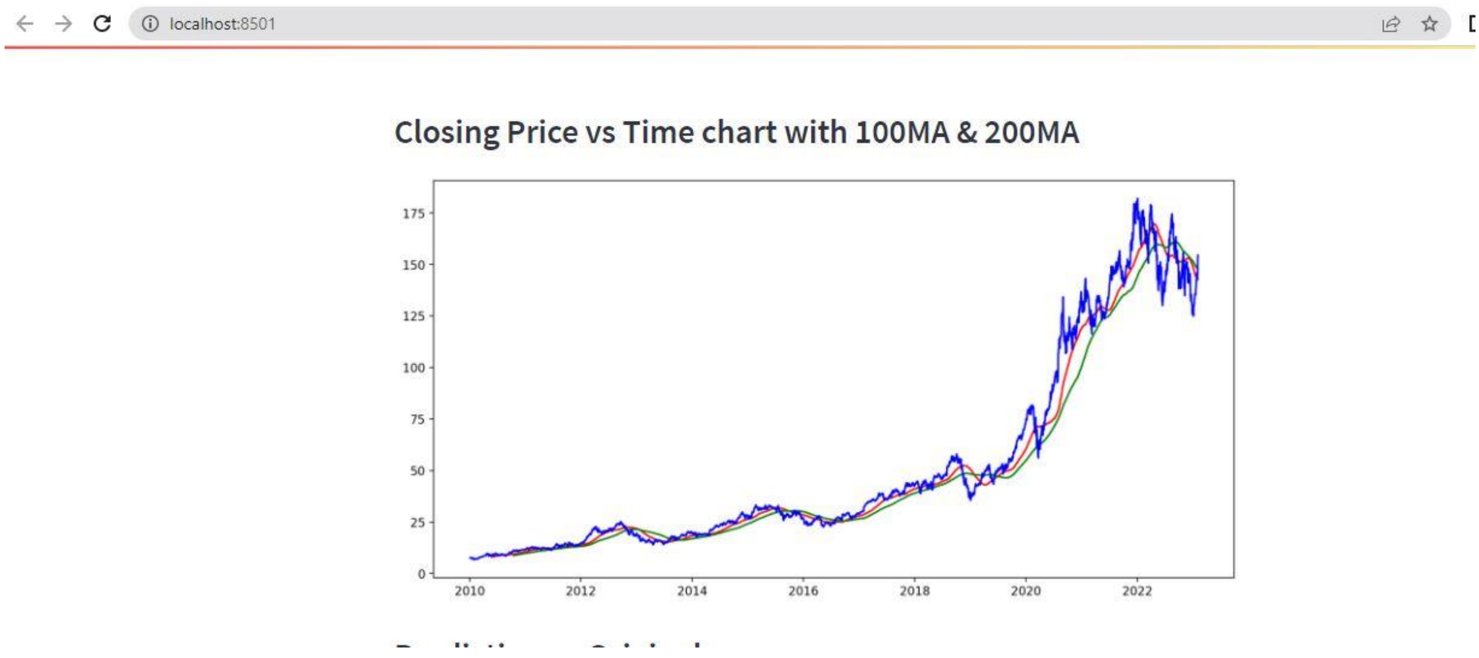


localhost:8501

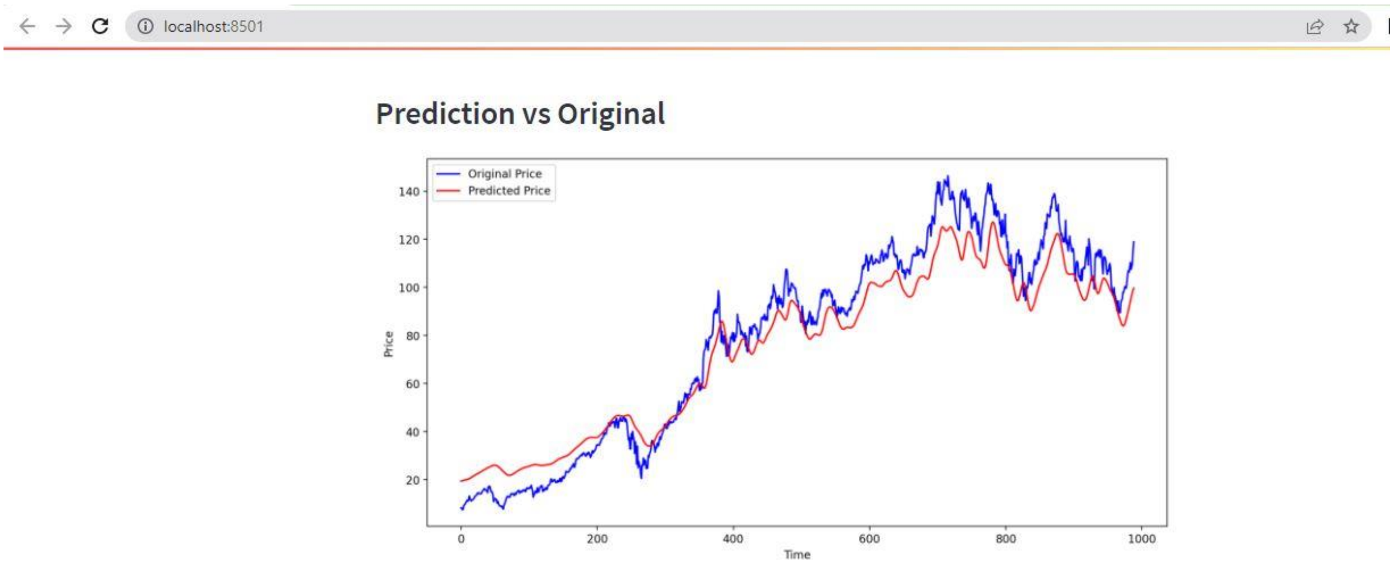
Closing Price vs Time chart



Closing Price vs Time chart with 100MA



Closing Price vs Time chart with 100MA & 200MA



Prediction vs Original

8. Reference

https://www.tensorflow.org/install

https://intellipaat.com/blog/what-is-lstm/#:~:text=LSTM%20stands%20for%20long%20short,especially%20in%20sequence%20prediction%20problems.

https://towardsdatascience.com/everything-you-need-to-know-about-jupyter-notebooks-10770719952b

https://www.javatpoint.com/python-yfinance-module